

Caligari2

Broadcast Rendering Module.

by Roman Ormandy

Copyright 1991 by Octree Software

Content:

1. Introduction	4
1.1 What is 3D Rendering?	5
2. Shading Tutorial	6
2.1 Shaders	8
2.2 Shading and Geometry	11
2.3 Lights	14
2.4 Shading Caveats	17
2.5 Texture Mapping	19
2.6 Environment Mapping	22
2.7 Shadows	25
2.8 Anti-aliasing	26
3. Basic Functionality	27
3.1 Broadcast Menu	29
3.2 Anim Menu	30
3.3 Lights Menu	32
3.4 Display Menu	36
3.5 Attribute Menu	47
3.5.1 Basic Attributes	50
3.5.2 Textures	62
3.5.3 Environment Mapping	65
3.5.4 Surfaces	69

3.5.5 Shaders	74
-------------------------	----

1. INTRODUCTION

With the release of the Broadcast module, Octree is filling the last missing piece in the Caligari product line, in its quest to deliver a full 3D professional design and video animation environment to a new class of users.

Up to now the achievement of full broadcast quality would required expensive specialized hardware and software, which often demanded a full-time operator undergoing several months of training. With Caligari, the situation changes. Not only is Caligari less expensive and more powerful, it also has an improved user interface which closely resembles the everyday 3D world in which we live. This type of user interface has been dubbed "Virtual Reality", and it will allow you to design and animate better and faster.

Broadcast Renderer adds to this user interface an image quality high-end applications demand: full color, smooth shading with anti-aliasing, Gouraud, Phong and metal shaders, texture and reflectance mapping, transparency, shadow generation, unlimited number of global or local lights, etc.

Caligari is not a ray tracer: its production oriented algorithms are 20 to 50 times faster while preserving the image quality today's user demands. At the heart of Broadcast Renderer is a rendering engine developed by Numerical Design's Turner Whitted: a true pioneer in the area of Computer Graphics.

1.1 What is 3D Rendering?

Despite the increasing popularity of 3D design and animation programs, the majority of computer systems in use by artists today are two dimensional, the most famous category being "paint" programs.

In a paint program, the user manipulates an image directly using a mouse or pen and redrawing pixels on the screen. In Caligari, the actual modification of screen pixels is outside of the user's direct intervention and is fully automatic and done by the program itself. While this may seem like a restriction of the artist's creative freedom, it allows the artist to concentrate on the actual composition of the scene, lights, and other optical considerations. And, of course, at any time a 3D rendered image can be loaded into a paint program for direct pixel manipulation.

The basic design process in 3D starts with the specification of objects chosen to compose the desired scene. The user specifies the geometry of the object, using a variety of Caligari design tools. At this stage, the user only sees a "wireframe" representation of the objects, but he can move them in full perspective, to see how they look from above or from the side. This process resembles the work of a sculptor.

The designed objects are then saved and loaded into the scene design, where the user composes the scene to be rendered and animated. While the initial composition is in wireframe, the user can always render the scene in progress using either Quick or Broadcast renderer. Quick renderer is just that - quick, for the initial rough setup of objects, colors, and lights in the scene. After the initial work, and when the scene is more defined, the user can switch to Broadcast Renderer for the fine tuning of the scene attributes.

A good analogy for a renderer is that of an abstract camera which can take snapshots of a world assembled together only by the user's imagination. In this world, impossible things can happen, objects can move through each other, metallic objects can be made transparent, and a person's face can be mapped onto a glass bottle.

Thus, instead of painting pixels on the screen, the user defines the geometry and surface attributes of the objects, assembles them into a scene, places the lights and viewpoint, and presto: Caligari will render a photorealistic image of the user's creation.

2. SHADING TUTORIAL

Surfaces, Lighting, and Shading

Shading at any point in a realistic computer-generated image is a function of surface properties, surface orientation, and the illumination of the surface. Surface properties include diffuse reflectance, specular reflectance, shininess (related to smoothness), transparency, and color. Illumination includes light source position, direction, orientation, color, strength, and distribution.

Caligari provides a rich set of methods for simulating the surface properties of objects. Associated with each surface in a scene is an attribute which specifies a model of reflection to be applied to the surface, along with the surface properties that augment the reflection model. In some cases, surface properties can be specified by a table to produce effects that vary across the surface. For example, a texture map can be used to modify the surface color and transparency, in effect mapping an arbitrary image onto a surface. (For more information on texture mapping, see the Texture Mapping Tutorial.)

2.1 Shaders

Caligari implements five different lighting models, called shaders, for a wide range of lighting effects. These are **Flat**, **Gouraud**, **Phong**, **Metal** and **Environment** shaders.

Flat

The flat shader implements no lighting model at all and merely copies the polygon's color and transparency to the display. It is useful for 2-D graphics and fast previewing. It is also great for preview of texture mapped objects and special effects with textures since the textures will look exactly like the pictures they were made of.

The remaining shaders are much more useful for producing realistic 3-D effects.

Gouraud

Gouraud shading provides a fast method of smooth shading diffuse (non-shiny) surfaces. In this method the correct color is calculated for each of the object vertices. Edges and inside of polygons are then simply interpolated among adjacent vertices.

The illumination of a Gouraud-shaded surface is controlled by two properties of the surface: the coefficient of ambient reflection, k_a , and the coefficient of diffuse reflection, k_d . k_a defines the amount of ambient light scattered by the surface. k_d defines the diffuse reflectance of the surface. Reasonable values for k_a and k_d are 0.375 and 0.625 respectively.

Phong

The Phong shader interpolates the surface normal across polygon interiors, then computes an illumination intensity independently for each pixel lying within the surface. The illumination consists of the sum of three components: ambient and diffuse light, as above, plus the specular light that is reflected by the surface from each defined light source toward the viewpoint.

In addition to the k_a and k_d coefficients as described above, Phong-shaded surface illumination requires two additional surface proper-

ties: k_s , the coefficient of specular light, and sp , the coefficient of surface shininess.

Using the Phong shader, reasonable values for k_a , k_d , k_s , and sp are 0.3, 0.4, 0.3, and 32, respectively.

Metal

The metal shader is loosely based on the Cook-Torrance shading model that is superior to the Phong model for simulating metallic surfaces. Because this shader is used to model highly reflective surfaces, the intensity calculation does not incorporate a diffuse lighting term. Furthermore, unlike plastic surfaces, on which highlights are simply reflections of the light source, metallic surfaces selectively absorb light, thereby producing specular reflections that vary between the color of the surface and the color of the light source, based on the angle of reflection. When a light source is reflected at a grazing incidence (such as near the silhouette edge of a backlit surface) the highlight takes on the color of the incoming light, whereas when the reflection is perpendicular to the surface, the highlight takes on the color of the surface.

Reasonable values for the ambient coefficient (k_a), specular coefficient (k_s) and shininess coefficient (sp) are 0.35, 0.65, and 12, respectively. The following table contains the recommended surface colors for several commonly-used metals:

Metal	red	green	blue	alpha
Bronze	165	.115	37	.255
Copper	211	.101	25	.255
Gold	255	.172	86	.255
Tin	249	.219	134	.255
Nickel	182	.155	96	.255
Stainless Steel	144	.177	107	.255

The metal shader is also an effective method for rendering glass surfaces. Whereas every shader can render transparent surfaces, only the metal shader will cause the transparency of a polygonal

surface to fall off near the edges of the surface, modeling the Fresnel effect.

Environment

The environment shader is a limited, 1-dimensional implementation of "environment mapping" that creates quite a reasonable approximation of surfaces such as chrome or glass.

2.2 Shading and Geometry

Surface Orientation and Normal Vectors

Because geometry and shading are so closely related, it is not fruitful to create a three-dimensional model without considering how it will be shaded.

A smoothly curved surface may be modeled by a mesh of polygons that approximates the position of the surface in space along with a set of normal vectors that indicate the orientation of the curved surface at each vertex of the polygon mesh. By interpolating these surface normal vectors across the faces of the polygons, it is possible to approximate the orientation of the original curved surface at each point within the polygon mesh. Based on these interpolated surface normals, it is possible to shade the polygon mesh so that it appears to be smoothly curved.

The resolution of the polygon mesh used to approximate a curved surface is an important consideration for the modeler. If the polygon mesh is too coarse (i.e. not enough polygons are used to model the surface) the faceted nature of the polygon mesh may be apparent in the renderings, both in subtle discontinuities in the shading of the mesh and in the visibly piecewise-linear silhouette of the mesh. If, on the other hand, the mesh is too fine, limited machine resources will be used to process an unnecessarily large number of polygons and vertices.

Facet Shading

The previous section describes how a surface consisting of a set of planar polygons can be shaded to appear to be smoothly curving. Using these techniques, a dodecahedron can be shaded to look like a sphere (although the geometrical shape will remain that of a dodecahedron, making it only an approximation of a sphere). The user may actually wish it to appear to be a dodecahedron with flat, not curved sides, and to have the lights present in the scene illuminate it as such.

To avoid these unnecessary expenses, Caligari allows the user to specify surfaces that are to be facet-shaded,

When faceted surfaces are encountered, the shading calculation (using the Phong, Gouraud, or metal shading model) is performed for each vertex of a facet.

The result is then averaged to produce a best approximation to the shade for the entire facet. During the rendering process, this color is applied to the entire facet, without the need for interpolation or recalculation of the shading function.

Simple facet shading does, however, have some limitations. First, textured surfaces cannot be handled. Shadows also cannot be handled since the relationship between the surfaces and the volumes of space that lie in shadow must be determined on a point-by-point basis. Thus, three forms of facet shading are supported: the simple form outlined above, in which the entire shading function is computed in preshade for use on non-textured surfaces that will not be affected by shadows. A more complex form, which will handle both textures and shadows is accessible through "complex facet" option.

Finally object like a beer can has smooth sides while the top and bottom is faceted. The "Auto Facet" option in Caligari will smooth out only those surfaces of an object which connect at a shallow degree (less than a user defined amount). Thus all sharp angles in the object will be preserved and not smoothed out.

Transparency

The transparency of a surface is defined by the alpha coefficient of its attribute. Whenever the alpha coefficient of a polygon is less than 255, the polygon is treated as partially transparent, with the degree of transparency proportional to the alpha value (0 is totally transparent and 255 is totally opaque).

As a transparent polygon is shaded, the shade from all surfaces behind it is filtered by the R, G, B, and alpha of the transparent polygon. In reality, the filtering of colored transparent surfaces is more complicated, but this approximation is a reasonable compromise. Note that a pure red filter in front of a cyan background will appear black.

Specular reflections on transparent surfaces are opaque. The opacity is proportional to the strength of the highlight. This model simulates the manner in which the reflection of a strong light source washes out other sources of shading.

If there is no opaque surface behind one or more transparent layers, then the combined alpha's of the transparent layers is written into the image file. This way, the transparency effect will be maintained when the image is merged onto a background. However, the color filtering effect will be lost since mergern, the compositing program, does not have enough information from the image data to determine the proper color filtering.

2.3 Lights

Illumination in the Caligari package is described by point light sources, environment maps, and an ambient light term. Point light sources that are infinitely far away from the scene are defined in terms of the direction from the scene to the light. Point light sources local to a scene are defined in terms of their position.

Diffuse reflection depends only on the light-source direction and the surface orientation. Specular highlights, on the other hand, reach a maximum value when the surface normal is aligned with a direction halfway between the direction from the point of reflection to the light source and the direction from the point of reflection to the viewer. In perspective views, the direction to the viewer changes from pixel to pixel, so that even if the light source is infinitely far away, the specular highlight will vary in amplitude across the face of a flat surface.

Although the ambient light is an illumination term, independent of the properties of any given surface, the reflection of ambient light has historically been specified as a constant that is attached to surface properties. Strictly speaking, this term is not constant and should be affected by the illumination terms (this is handled properly by the Radiosity method).

Environment maps provide a better simulation of ambient light. For very shiny surfaces, the ambient light passed to the viewpoint from a point on the surface is actually a reflection of the environment of the surface.

Details of environment mapping are given in the Environment Mapping Tutorial.

Positioning Light Sources

Although in Caligari the distance between a surface and the light source doesn't affect the illumination, the actual direction does. For this reason, very different effects can be achieved by using local light sources rather than infinitely distant sources. The illumination of a point on the surface of a scene is determined from the orientation of the surface at that point (the normal direction) and the direction toward the light source. The diffuse light striking the surface is most intense when the surface is facing the light exactly; the

specular reflection of the surface is most intense when the light is reflected toward the viewpoint by the surface.

When infinitely distant light sources are used, the direction toward the light from every point in the scene is constant. Thus, every point on a planar surface will receive exactly the same diffuse light (although potentially different specular light, since the direction toward the viewer changes across the surface). Because the light direction is constant, much of the lighting calculation can be done just once for the entire polygon which makes the shading calculation very fast.

When the light source is local to the scene rather than infinitely distant, the direction toward the light can change across a planar surface.

If a light source is placed just in front of the center of a large square surface, only the center of the square surface actually faces directly toward the light source and receives the most diffuse light. Unfortunately, the calculation to determine the direction of a local light is very time-consuming; thus, images containing local lights will take substantially longer to generate. It should be noted that much of this cost is incurred only once for each illumination calculation; the marginal cost of further local light sources is relatively low.

Directional "Spot" lights are like local lights, but they cast light in one direction in a conical shape much like the beam of light cast by a flashlight. The computation time for spot lights is even greater than that for local lights, and unlike local lights, each additional spot light in a scene adds a proportional amount of computation time. Spot lights should therefore only be used when absolutely necessary to achieve a particular lighting effect.

Pitfalls

A problem arises from trying to model concave, non-planar polygons. Here, the interpolation of a surface normal across the face of the polygon is highly dependent on the orientation of the polygon. That is, in an animated sequence, the shading of the polygon would vary in an inconsistent manner. The best advice in this case is to break concave, non-planar polygons into pieces which are either planar or convex.

Many geometric modeling packages produce only triangles (which are always both convex and planar) to avoid shading problems. However, Caligari is designed to deal with concave planar

polygons, and designers should not hesitate to create them as long as the above conditions are not violated.

Subtle shading problems can occur when Gouraud shading is mixed with local light sources. It is best then to use Phong and Metal shaders exclusively in scenes that contain local light sources.

2.4 Shading Caveats

To maintain high performance, the number of light sources available with Gouraud shading is limited to three. A user who wants to shade a diffuse reflector with more than three light sources should instead use Phong shading, with the specular reflectance coefficient, k_s , set to zero.

One should note that a diffuse surface illuminated by more than three lights probably will look a little washed out. Gouraud shaded surfaces are not compatible with spot lights; they are treated as local lights and have the same limitations.

The flat and environment shaders do not support facet shading. Using simple facet shading with the Gouraud, Phong, and metal shaders will disable texture and environment maps, as well as shadows. (Complex facet shading can be used to preserve these effects.)

The five different shaders respond differently to lighting and mapping effects. For example, the flat shader does not support shadows (a lighting effect) since it does not respond to lights. The Gouraud shader does not support environment mapping because there are no specular reflections. The shaders silently ignore requests for effects that they don't support.

The following table summarizes which shaders support which lighting and mapping effects:

	flat	Gouraud	Phong	metal	environment
Infinite lights	. . . b . . a . a				
Local lights	. . . b . . a . a				
Spot lights	. . c . . d . d				
Texture Maps	e . . f . . . f . f				
1-D Env Maps	 f . f . . . g . .				
2-D Env Maps	 f . f				
Shadows	. . . f . . . f . f				

- a) Supported in all modes.
- b) The Gouraud shader is limited to a total of three light sources. Either infinite, local, or a combination of both may be used. If more than three light sources are specified, only the first three will be used.
- c) A directed light is treated the same as a local light by the Gouraud shader.
- d) A directed light is treated the same as a local light by the Phong and metal shaders in simple facet shading mode.
- e) The flat shader does not support facet shading; therefore texture maps are only supported in the non-facet shading mode.
- f) Ignored in simple facet shading mode.
- g) The environment shader does not undo the viewing transformation before applying the 1-D environment map; i.e., the reflection is based solely on vertical elevation in the eye coordinate system.

2.5 Texture Mapping

In computer image generation, texture mapping is used to increase the visual complexity of an image without a corresponding increase in geometric complexity. Mapping a texture onto a surface is much like applying a label or decal onto an object. Unlike physical labels or decals, however, computer-generated textures can be stretched and distorted to conform to a surface exactly. An efficient method for creating a convincing brick wall is to apply a "brick" texture to a single polygon; this produces the desired effect without explicitly modeling each brick in the wall. A realistic-looking wine bottle can be created by mapping a label onto a bottle shape. Applying the label with texture mapping alleviates the need to re-render the detailed label each time the wine bottle is used in an image.

The correspondence between the texture map and a surface is determined by the values of two surface parameters, u and v .

For each output pixel, the (u,v) indices of the corresponding surface point are used to find the color (r, g, b, alpha) in the two-dimensional texture map.

The modeler specifies the assignment of u and v parameters to the surface by explicitly defining their values at each vertex of the surface. Consider a flat rectangle. Each corner of the rectangle is assigned a pair of (u,v) values: $(0,0)$ at the lower left, $(0,1)$ at the upper left, $(1,0)$ at the lower right, and $(1,1)$ at the upper right. During the rendering process, these vertex values are interpolated across the rectangle to produce a (u,v) coordinate pair for each interior point; these are then used to access the texture map to produce a color for the point.

More interesting effects result by warping the textured surface into more complex shapes. One such deformation might be to roll the textured surface into a cylinder. The texture remains attached to the surface without undo distortion; the u parameter varies around the circumference of the cylinder, and the v parameter varies along the length of the cylinder.

The effective use of textures requires an understanding of the preparation of the texture map, as well as the geometric model on which a texture is to be mapped.

Caligari's mapped textures have some limitations. Objects created with lathe can not be mapped if they are rotated less than 360

degrees and also if they are not rotated around the edge of the lathed polygon. Extruded shapes also cannot be mapped in this way. All objects however can be textured using the projection method.

Preparation of Geometric Models for Texturing

Preparing a model for the application of a texture involves both the specification of the geometric mapping of the texture onto a surface and the specification of the shading attributes of the textured surface.

Surfaces may be warped into any shape without changing the mapping of the texture onto the surface. In effect, the texture is "pinned" to the surface by the assignment of the texture to the vertices of the mesh regardless of the warping. Thus a planar mesh may be warped into a cylinder, with the right- and left-hand edges of the texture meeting where the edges of the surface meet.

Scaling and offsetting are specified in terms of the unit square in texture space. To reduce a texture in size, the user chooses scaling parameters (in u and v) corresponding to the fraction of the width and height of the unit square to be covered by the texture. (The entire unit square in texture-map space is mapped onto the destination surface.) Similarly, the texture offset is chosen in terms of proportions in the unit square. Thus, a texture one-half the width and height of a surface may be mapped onto the center of the surface by scaling the texture by values of 0.5 and 0.5, while offsetting the texture by 0.25 and 0.25.

The Caligari display system also allows the use of repeated textures. When using the repeat texture option, the unit square in the texture space is covered by "copies" of a scaled-down texture. Thus, if we scale a texture to one-fourth size and specify the repeat texture option, the resulting textured surface will be covered by 16 copies (4 rows of 4 columns) of the texture.

Specifying a Textured Surface

The texture name supplied in the texture-table attribute specifies the name of the file containing the texture map. When a file name is given, the "CalTexture:" directory is searched for that file.

Anti-Aliasing in Texture Maps

To remove visible artifacts on texture-mapped surfaces, Caligari provides texture map anti-aliasing. Aliasing in texture mapping occurs when the texture map is sampled too infrequently to reflect the detail in the sampled texture accurately. Caligari solves this problem by using a pyramidal filtering algorithm similar to MIP-mapping. As part of the conversion of an image to a texture map, a set of pre-filtered versions of the texture is created, with each successive version filtered and stored at half the resolution of the previous level. During the rendering process, the texture map sampling rate is determined by finding the dimensions of the pixel square in the texture space.

The user has the option to turn the anti-aliasing of textures off.

2.6 Environment Mapping

Techniques

Most shading models simulate the reflection of point light sources from surfaces. In the real world, shiny surfaces reflect objects around them as well as simple light sources. Recursive ray tracing is a well-known but very time consuming method of simulating the reflection of one object in another. Environment mapping is a slightly less accurate but more efficient technique for accomplishing essentially the same result. (Ray tracing, on the other hand, simulates other effects besides reflection of the environment.)

An environment map is an image showing the world as seen from the surface of a reflecting object. Clearly, the viewpoint should be different for each point on the object, but, in practice, a single viewing position works well, as long as reflected objects are far away.

Unlike most images, the environment map must wrap completely around the scene at its center. Environment maps can be conveniently regarded as an image projected onto the surface of a sphere of infinite radius surrounding a reflecting object.

In Caligari, two slightly simpler formats are used to store environment maps. In many outdoor scenes, the reflected environment consists of little more than the horizon and the sky. In this case, the 2-dimensional environment map varies only in the vertical direction and can be effectively replaced by a 1-dimensional table. For the more complex environment of indoor scenes, a 2-dimensional map is necessary. However, Caligari uses a six sided cube instead of the spherical map. Not only is the six sided map simpler to create, but the program that computes the look-up table indices is simpler and faster than one for a spherical map.

Reflection of Simple Environments

Imagine a mirrored sphere suspended above the floor of a perfectly flat desert on a cloudless day.

Assume as well that the sun can be represented by a point light source which is not part of the immediate environment. Because the reflected environment changes only with elevation angle, the reflected colors can be stored in a one dimensional table.

While this simple form of environment map provides surprisingly realistic results, it doesn't work in all cases. For example, if the example just cited were to include clouds in the sky, a two dimensional environment map would be required.

Reflection of Complex Environments

Real scenes are rarely simple. Indoor scenes, for example, comprise environments which are both rich in detail and not uniform in any direction.

Shading with Environment Maps

The Caligari Phong and metal shaders support either one or two dimensional environment mapping.

While the one dimensional map is nothing more than a special case of the two dimensional map, the shaders can be made much more efficient if the two cases are distinguished.

Caligari includes a special shader, the environment shader, which is used exclusively for one-dimensional environment mapping. Since it reflects the environment totally, it is only useful for simulating chrome-like surfaces. Its advantage over other shaders is speed. Since the overhead of accessing two-dimensional environment maps negates most of the speed advantage of the special shader, chrome-like surfaces which reflect two-dimensional environments should use Phong shading with a color of 255,255,255, and reflection coefficients $k_a = 0.0$, $k_d = 0.0$, and $k_s = 1.0$.

Both the Phong and metal shaders can incorporate the environment into their specular components. Unlike the totally reflecting environment shader, these shaders reflect selectively. For example, a shiny gold surface will absorb most of the blue light in the environment.

This ability to simulate a variety of surface finishes, not just chrome, provides additional utility to Caligari shaders using environment mapping. Furthermore, the use of environment maps does not preclude the use of other features, such as texture mapping (except in the environment shader which doesn't support texture mapping anyway).

Since the flat and Gouraud shaders are not useful for simulating shiny surfaces, it makes no sense to incorporate environment mapping into them. Environment mapping is disabled for these shaders.

2.7 Shadows

Caligari simulates shadows using a shadow volume technique. This tutorial describes the principles of the technique, describes the additional display pipeline element needed to implement shadows, and provides examples.

Shadow Volumes

Caligari uses a shadow volume algorithm to compute shadowed regions of an image. While other approaches such as ray tracing or z-buffering may be simpler, the shadow volume algorithm is generally better suited for use with scan line algorithms.

A shadow volume for a polygonal object is defined as the volume swept out by the silhouette of the object in the direction away from the light source (together with the volume of the object itself).

Any point on the interior of the shadow volume will be in shadow with respect to that light source.

Shadow Effects in Caligari

In Caligari's shading model, shadows are modifiers of light source color and intensity. The most obvious shadow effect is the complete blocking of light by an opaque shadow casting object.

Objects which are not completely opaque, however, cast shadows as well. A green semi transparent bottle would cast a greenish shadow if illuminated with a white light.

Caligari's model of the light transmission properties of translucent shadow casting objects is simplistic. The shadow effects do not account for changes in light transmission due to changes in orientation of shadow casting polygons, nor do they account for texture maps applied to a shadow casting object. For example, if the sphere were completely transparent with an opaque texture mapped onto it, no shadow would be cast.

Or, if a green glass wine bottle had a label texture mapped onto it, only a green bottle-shaped shadow would appear. No shadow would be cast by the label.

2.8 Anti-aliasing

Since aliasing in computer generated images comes in several different forms, Caligari combats these artifacts with several different mechanisms. The challenge of doing anti-aliasing in a small computer with limited processor speed and memory capacity has forced some compromises in trading off efficiency with effectiveness.

However, in almost every case, Caligari provides a path for producing images of arbitrarily high quality at the expense of added processing time.

Since Caligari uses a scan line ordered rendering program, it is a simple matter to filter analytically in the horizontal direction. While it is theoretically possible to filter analytically in the vertical direction as well, the implementation of Caligari actually oversamples in the vertical direction; i.e., it calculates more scan lines than it actually writes to the display, and filters the result. Nearly horizontal edges will look better as vertical oversampling rates are increased.

While the time required to render a scene doesn't increase quite linearly with the amount of vertical oversampling, it is a fair rule of thumb to use linear cost.

Specular Aliasing

Not all aliases in synthetic images appear along edges. In regions of high curvature on glossy surfaces, the renderer may produce a very narrow highlight. If the highlight is sufficiently narrow, it will exhibit the same "staircasing" artifact seen in an unfiltered edge. An analogous artifact appears in environment maps which contain high frequency features.

3. BASIC FUNCTIONALITY

The menu structure of Broadcast Renderer is consistent with other Caligari modules. The user enters Broadcast Renderer through the "BRender" button on the scene menu.

The user can render the whole scene in full color simply by selecting "Render" from BRender menu.

Important!

At any time, the rendering process can be interrupted by the user by pressing the Control-C combination on the keyboard.

Caligari is not a ray tracer. It is more than 20 times faster than comparable Amiga ray tracers and it is the fastest renderer on the market. Still, a large scene will require several megabytes of RAM and may take upward of 15 minutes to render. When rendering several hundred frames for a video animation, time for rendering each frame is critical. There are a few things the user can do to speed up the rendering process.

The first one is to get the most RAM and the fastest processor money can buy. This means 5 to 20 Mb of RAM and 68030 and 68882 or 68040 CPU with the fastest clock cycle. The initial hardware outlay may be expensive, but it will pay for itself with increased productivity.

Secondly, the user can change the default rendering from a hard disk to a RAM drive. Caligari uses the "CalScratch:" directory during rendering as for intermediate storage. If there is enough RAM available (more than 5Mb) the user can tell Caligari to use it instead of a hard disk. This can be done by un-commenting the appropriate line in the "Caligari-Setup" script file.

Purpose:

To enter the Broadcast Renderer submenu.

Procedure:

1. If you are not in the Scene Module, enter it now.
2. Select "BRender" from the menu.
3. Broadcast submenu will appear.

3.1 Broadcast Menu

The "Broadcast" menu presents itself initially as a single strip at the bottom of the screen. This is the base menu for all operations involving Broadcast Renderer. The arrangement of the buttons on the menu is as follows:



Framebuffer - Allows the user to change the frame buffer to be accessed for rendering.

Anim - Enters the Script module for both the real time preview and final full-color animation to be saved to disk.

Lights - Allows the user to specify lights for Broadcast Renderer.

Display - Sets global display options such as the size of the output picture, level of anti-aliasing, etc.

Attr - Displays menu for assignment of broadcast attributes to an object.

Images - Allows the user to display full color images that have been previously saved.

Render - Renders current scene in full-color using the selected frame buffer.

Scene - exits Broadcast Renderer into the Scene Composition module.

3.2 Anim Menu

The "Anim" menu is identical to the one in the quick renderer. The major difference is in the ability to save all of the rendered images comprising the animation to disk. This eliminates the need to use a single frame controller for immediate recording to the VTR. Additionally, the user can select a script to be loaded in a manner consistent with the rest of the program, using the standard File Requester.

Caligari can save (using the "Save" option) all the frames generated by a script to the directory where the script resides.

Purpose:

To render frames from current script and save them to the disk.

Procedure:

1. Select "Anim" from Scene or Broadcast menu. Select "I/O".
2. Load a script. Select "Save" from the AnimI/O menu; then select "Render". The script will be rendered frame-by-frame, and each frame will be saved to the script directory.

Note:

You can save rendered animations in Quick or Broadcast renderer. After saving the images, they can be processed by other third party software and compressed into an animation that can be played back in real time, converted to another image format which may be more suitable for some applications, archived on disk and sent to a service to be recorded on tape frame-by-frame, etc.

3.3 Lights Menu

The "Lights" menu allows the user to light the scene in a manner similar to a "real" movie set. It is here where the results of a design and animation effort can be ruined by poor lighting or improved with a well-lit scene.

Add Light	Light <<0>>	HO	r255	Inside	255	Hide	Exit Lights
	Infinite	80	0255	Outside	0	Red sun	
	Local	U	0	0255	88.8	Narrow	
	Spot	From	x -28.888	To	x 0.888		
Delete	Shadow	Eye	y 88.888	Object	y 0.888		
		z 100.000	z 0.000				

The Lights menu is, again, similar to the one in the Quick renderer. However, there are some differences. Essentially, the Broadcast Lights menu is a superset of the Quick renderer menu, and it is upwardly compatible with it. This means that if the user sets the lights initially in the Quick renderer and then switches to Broadcast Renderer, the configuration of lights will be preserved. On the other hand, if the lights are further modified in Broadcast Renderer, upon return to Quick render, some information may be ignored.

The major difference is the ability of the user to specify local and spot lights and give each light a true color (not from a lookup map) and direction.

For each light the user can also decide whether or not it will cast a **shadow**.

Purpose:

To make the selected light a local light.

Procedure:

1. Bring up the "Lights" menu from the Broadcast base menu.
2. Select "Add". Set color and X, Y, and Z values for the new light.
3. Select "Local".

Note:

Unlike for global lights, X, Y, and Z define the actual point where the local light is located. For a Global light X, Y, and Z are only a vector (defined by X, Y, Z and 0, 0, 0) since the light itself is infinitely far away. (See tutorial)

Tip:

A good way to place the lights is to position the viewpoint to where the light should be (be sure that the Eye Scale slider is centered) and use the "From EYE" option on the menu.

Purpose:

To make a directional light with a cone shade.

Procedure:

1. Bring up the "Lights" menu from the Broadcast base menu.
2. Select "Add". Set color and X, Y, and Z values for the new light.
3. Select "Spot" and size of the cone (Wide, Medium or Narrow). If one of the default cone size values is not sufficient any value between 0 and 90 degrees may be entered into the "Cone" field.
4. Set the light direction with the second pair of X,Y and Z coordinates next to the "To Object" button or select an object and click on "To Object".
5. Optionally select the light intensity inside and outside of the cone (0 = minimum, 255 = maximum).

Purpose:

To request that a light cast shadows.

Procedure:

1. Select "Lights" from base Broadcast menu.
2. Select "Shadow". Next time the user renders the scene, the selected light will cast shadows.

Note:

The use of shadows enhances considerably the realism of the rendered image. Rendering time will also increase, so use the shadows only in the final stages of production and where absolutely necessary.

Users should note that flat polygons have a light side and a dark side. Shadows cast on a polygon's dark side will not be visible. Therefore, flat polygons should be rotated so that their dark sides are facing away from the shadow-casting object. When in doubt use a flattened extruded object instead, although the shading is not identical.

Example:

- Design a scene composed of a flat horizontal polygon and sphere hovering above it.
- Use the default light and select "Shadow" from "Lights" menu. Select "Render" and you should see the sphere cast a shadow on the flat polygon beneath it. If the shadow does not appear, then rotate the flat polygon 180 degrees in either the X or Y axis.



3.4 Display Menu

The "Display" menu allows the user to set display parameters such as resolution, level of anti-aliasing, etc. These are scene-independent and must be set by the user prior to rendering. Access to the Display menu is from the base Broadcast menu.

Aspet	Vid	NTSC	PAL	512	x	485	Mode	Solid	Size	1/1	A-Alias	Horiz	Other
	Background		Load		Foreground		Load	Save					

☐ ☐

Purpose:

To set the output image size to one of those compatible with the NTSC video standard for the selected frame buffer.

Procedure:

1. Bring up the "Display" menu from the Broadcast base menu.
2. Click the "NTSC" button. Each time the user clicks on the NTSC button, a new resolution will appear until the list of resolutions available for the selected frame buffer is exhausted, at which time it starts again at the beginning.

Purpose:

To set the output image size to one of those compatible with the pal video standard for the selected frame buffer.

Procedure:

1. Bring up the "Display" menu from Broadcast base menu.
2. Click the "PAL" button. Each time the user clicks on the PAL button, a new resolution will appear until the list of resolutions available for the selected frame buffer is exhausted, at which time it starts again at the beginning.

Purpose:

To set the output resolution to be full, half or quarter screen size.

Procedure:

1. Bring up the "Display" menu from the Broadcast base menu.
2. Select "1/1", "1/2" or "1/4".

Note:

Caligari can simultaneously display up to five quarter size images and one half size image. This is useful both for quick previewing of the final image and comparison of different color schemes or light compositions for the scene.

Purpose:

To render the scene with all the attributes.

Procedure:

1. Select "Display" from the base Broadcast menu.
2. Select "Solid".

Note:

This is the default configuration for the display menu.

Purpose:

To render the scene with all objects having a faceted appearance.

Procedure:

1. Select "Display" from base Broadcast menu.
2. Select "Facet". Next time the user renders the scene, all the objects will have a faceted appearance.

Note:

The primary purpose of the "Facet" display option is a quick preview of the scene.

Purpose:

To render the scene in color wireframe.

Procedure:

1. Select "Display" from the base Broadcast menu.
2. Select "Wire". Next time the user renders the scene, all the objects will be displayed in wireframe.

Note:

This is a fast renderer, but it needs quite a bit of memory while working. You may want to the the null framebuffer while rendering a wireframe image and then select the framebuffer for display and load the image. The wireframe renderer can be useful for special effects and previewing.

If anti-aliasing is set to any mode other than "None", then the scene will be rendered as an anti-aliased wireframe.

Purpose:

To avoid a banding effect in the image by blending the colors together.

Procedure:

1. Select "Display" from the base Broadcast menu.
2. Select "Dither". Render the scene to see the effect of dithering.

Purpose:

To remove jaggies and other artifacts from the image.

Procedure:

1. Select "Display" from the base Broadcast menu.
2. Select "None", "Horiz", "Full" or "Max". Render the scene to see the effect of anti-aliasing.

Note:

Anti-aliasing improves substantially the appearance of the rendered image by smoothing color transitions between adjacent pixels. Keep in mind that a better looking image will take longer to generate; thus, use only as much anti-aliasing as necessary.

"None" - Default. Means no anti-aliasing at all.

"Horiz" - Horizontal anti-aliasing. This option will smooth jaggies on each scanline as it is rendered, consuming only a little extra time as compared to no anti-aliasing at all.

"Full" - This option will smooth the jaggies in a vertical and horizontal direction producing better results. This option increases further the time to render the image.

"Max" - Serious antialiasing.

Example:

- Design a scene and select a view with some edges being almost (but not completely) horizontal.
- Render the view with Anti-aliasing set to "None". You should see lots of "jaggies" - a staircase like effect.
- Render the view with Anti-aliasing set to "Horiz" and "Full" and watch how the image improves each time.

ASPECT RATIO

Purpose:

To create images which have the correct ratio of pixel height to width.

Procedure:

1. Select "Display" from the base Broadcast menu.
2. Select "Vid" or "1:1".

Note:

Most images should be generated with the Vid aspect ratio, because this will create images with an aspect of 4:3 which is correct for NTSC or PAL video output. Some computers and image processing equipment (e.g. slide recorders) work with square pixels which have the 1:1 aspect ratio. The image which appears on any of the Caligari compatible frame buffers will appear the same with either setting, but the aspect ratio will be encoded into the image and other software which is used to process the image may need the correct aspect ratio.

Foreground and Background

Purpose:

To merge images "behind" or "in front of" the 3D rendered image.

Procedure:

1. Select "Display" from the base Broadcast menu.
2. Select "Background", "Foreground" or both.

Click on the "Load" button associated with the Foreground/Background button(s) previously selected.

The standard file requester will appear listing all of the .6rn images in the current directory. Select one of the .6rn files from this or any other directory.

Result:

After rendering an image, first the background (if selected) image will be "merged" with the rendered image, and then the foreground will be merged with the result of the previous merge. What this allows you to do is: draw an image in a paint program and use it as a background for the 3D scene, render a very complicated static environment once and use it as a background for an animation, superimpose previously painted or rendered objects on top of a 3D rendered scene, or any combination of the above.

Note: when merging images of different sizes, the resulting size of the image after the merge will be that of the background (e.g. if merging a 640x400 background with a 320x200 3D image and a 400x400 foreground, the result will be 640x400, but if just the 3D image and the foreground were merged, the result would be 320x200).

3.5 Attribute Menu

After selecting Brender from the Scene module and Attr from the Brender submenu an attribute menu is displayed.

From this menu the user can assign a variety of advanced rendering attributes to every material of an object in the scene and see the results by rendering the object, a subpart of the object or the whole scene.

Hue	231	R	0.231	Ambi	0.375	Stapler	10	Load Env
Sat	6137	G	0.625	Diff	0.625		20	Make 2D
Val	8115	B	0.000	Shin	0.000	Environment		View
Trn	255	A		Material	1	Persp	Top	Smooth
Render	Object	Sphere	Material	Load	Save	Side	Front	Gouraud
								Controls

Hue - Horizontal slider for selection of hue (color) assigned to the object

Sat - Slider for selection of saturation (color purity) assigned to the object.

Val - Slider for selection of value (color intensity, lightness) assigned to the object.

R, G, B - Numerical entries for specifying the color in RGB mode.

Tran - Specifies the level of transparency of an object. It has a numerical entry field to the right of the slider.

Amb - Specifies the level of ambient lighting.

Diff - Specifies the ratio between the diffuse and specular coefficients.

Shin - Specifies the specular power (shininess; size of specular highlights) of an object.

Spec - Specifies the amount of specular reflection.

Textures

Texture - Assign loaded texture to selected attribute.

Controls - A submenu for positioning textures on the surface of an object. It includes the Xoff, Yoff, Xscale, Yscale, and Repeat buttons as well as the tools to convert images to texture maps.

Environments

1D, Cubic - assign a one or two dimensional environment map to the selected attribute.

Load Env, Make Cube - Load 1 dimensional or generate a Cubic environment map.

View Env - View an environment

Surfaces

Smooth, Simple Fct, Complex Fct, Auto Fct - Selections for appearance of the surface of the object; smooth, faceted with one color per polygon, faceted with complex light model and automatic smoothing with preservation of sharp edges.

Shaders

Flat, Gouraud, Phong, Metal, Environ - Selections for the light model to be used in the rendering of an object.

Render Object - Render selected object in full color with current attributes.

Render Material - Render selected material in full color.

Render Sphere - Render selected material in full color as a high resolution sphere.

Persp/Top/Front/Side - for selecting the eye point.

Additionally, the attribute menu displays the name of the selected object, and the name of the environment (if any) assigned to the current material attribute. The user can change the number of the material attribute to be modified ("Material"). Each object can have a number of material attributes assigned to it in Object Design module.

Material attributes (sometimes called color attributes in Quick Renderer) are assigned as different colors to any subobject of an object in Object Design. The user sees the current attribute highlighted in an approximated color in the wireframe design space.

Thus, a car can be colored in object design with all wheels having material attribute (color) 1, and the body material attribute 2.

In the Attribute menu, the user can select the car and assign a texture to material 1 and a metallic blue color to material 2. He can then select from the scene another copy of the same car and assign different material attributes to it. It is not possible, however, to change the color of each wheel individually. To do that, the user would have to go back to Object Design and assign to each wheel a separate material attribute.

Note: objects can also be picked in the Attr menu by typing their name into the name field at the top of the Attr menu and pressing Return. If named object is in the workspace, it will be selected. This is useful for picking very small or large objects and objects which are not currently on the screen.

3.5.1 Basic Attributes

There are a few sliders at the left side of the Attr menu. These are basic attributes specifying the appearance of an object.

Purpose:

To assign color (hue) to an object.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Drag the slider labeled "Hue" to the left or right.

Note:

In order to see the effect of the new color, you must render the object. Unlike in the QRender, there are not any lookup tables, and the user simply assigns absolute color to the selected object.

Any change of hue (or saturation, or value) will immediately update the numerical values for RGB to the right side of the sliders (conversely typing in RGB values changes slider positions) and change the wireframe color of the current material. Since RGB and Hue/Sat/Val are different models for descriptions of color, the relationship between the slider position and numerical value will not be proportional.

Example:

- Select a simple object and render it in full color.
- Bring up the Attr menu and drag the "Hue" slider to the left and render again. This time the color of the object will be different.



Purpose:

To change saturation (color purity) of color of selected object.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Drag the slider labeled "Sat" to the left or right.

Note:

A fully saturated color (value of 255) is a pure color pigment. A decrease in saturation makes the color more pastel-like. No saturation (value 0) creates a shade of gray.

Example:

- Select a simple object and render it in full color.
- Bring up the Attr menu and drag the "Sat" slider to the left. Render the object again. This time the color of the object will be a little "washed out".

Purpose:

To assign intensity (lightness) to an object.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Drag the slider labeled "Val" to the left or right. Alternatively, type in a value from 0 to 255 in the numerical entry next to the slider.

Note:

An intensity of zero defines a totally black object.

Example:

- Select a simple object and render it in full color.
- Bring up the Attr menu and drag the "Val" slider to the left (not all the way!). Render the object again. This time the object should appear darker.



Purpose:

To specify the color in RGB mode.

Procedure:

1. Bring up the Attrb menu, select an object, then select one of its material attributes.
2. Type in a value from 0 to 255 in the numerical entry next to the R, G, and B label.

Note:

RGB mode means that each color is defined by varying amounts of red, green, and blue primary colors. It is possible to define every color from these primary components.

Any change of hue (or saturation, or value) will immediately update the numerical values for RGB to the right side of the sliders and the wireframe of the current material (conversely typing in RGB values changes slider positions). Since RGB and Hue/Sat/Val are different models for descriptions of color, the relationship between the slider position and numerical value will not be proportional.

Example:

- Select a simple object and render it in full color with the default attributes.
- Bring up the Attr menu and type 255 next to R, 172 for G, and 86 for B. Set the Metal shader and render the image. The above values define a gold color (see tutorial).

Purpose:

To assign transparency to an object.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Drag the slider labeled "Tran" to the left or right. Alternatively, type in a value from 0 to 255 in the numerical entry next to the slider.

Note:

In order to see the effect of the change in transparency, you must render the object.

The default transparency is 255. This defines a completely opaque object. As the values decrease, the transparency increases. The value 0 defines a completely transparent (invisible) object. However, even a completely transparent object will exhibit specular highlights and Fresnel effects if using the Metal shader.

Example:

- Design a scene with two simple objects, one in front of the other.
- Select the front object and render it with full opacity.
- Bring up the Attr menu and drag the "Tran" slider to the left. Render the object again. This time, the background object should be visible behind the front object.
- Next change the Saturation or Value of the color and rerender the object again (with the same level of transparency). Observe the changes.



Purpose:

To assign a level of ambient reflection to a material.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Drag the slider labeled "Amb" to the left or right. Alternatively, type in a value from 0.0 to 1.0 in the numerical entry next to the slider.

Note:

In order to see the effect of the change in the ambient coefficient, you must render the object.

The ambient coefficient defines the amount of ambient light scattered by the surface of the object. Ambient light is defined as indirect light or light reflected from the environment, rather than arriving directly from the light source. Traditionally, this has been a constant (and it is this constant that you are setting).

Indirect light, however, can often play a large role in the lighting of the scene. Caligari provides for a better approximation of ambient light in its environmental shader, which for very shiny surfaces replaces the ambient term with a reflection of the environment surrounding the object. (see Environmental shader)

Example:

- Select a simple object and render it in full color.
- Bring up the Attr menu and drag the "Amb" slider to the left. Render the object again. This time the edges of the object which are not reached directly by light should appear darker.

Note: when entering numerical values, you are not restricted to the range of 0.0 to 1.0. Traditionally the sum of the Ambi, Diff and Spec

values total 1.0, but we do not impose this restriction. Feel free to experiment with values outside this range, but we do not recommend that you exceed a total of 2.0 because you will begin to introduce artifacts and generally undesirable effects into the image. If the Ambi, Diff, Spec total does exceed 1.0 and you then use a slider to change one of the values, they will ALL be recomputed to bring the total back to 1.0.



Purpose:

To set the diffuse reflectance of a material.

Procedure:

1. Bring up the Attrb menu, select an object, then select one of its material attributes.
2. Drag the slider labeled "Diff" to the left or right. Alternatively, type in a value from 0.0 to 1.0 in the numerical entry next to the slider.

Note:

The diffuse coefficient defines the diffuse reflectance of the surface. An object with a high degree of diffuse reflectance, such as chalk absorbs incoming light and re-emits it equally in all directions. Thus, the color of diffusely reflected light takes on the color of the object's surface.

Example:

- Select a simple object and render it in full color with default coefficients, but make sure you select "Phong" from the shaders on the right side of the screen.
- Bring up the Attr menu and drag the "Diff" slider to the right. Render the object again. This time the object should have more pronounced highlights.

Note: when entering numerical values, you are not restricted to the range of 0.0 to 1.0. Traditionally the sum of the Ambi, Diff and Spec values total 1.0, but we do not impose this restriction. Feel free to experiment with values outside this range, but we do not recommend that you exceed a total of 2.0 because you will begin to introduce artifacts and generally undesirable effects into the image. If the Ambi, Diff, Spec total does exceed 1.0 and you then use a slider to change one of the values, they will ALL be recomputed to bring the total back to 1.0.

Purpose:

To assign a level of specular power (shininess) to a material.

Procedure:

1. Bring up the Attrb menu, select an object, then select one of its material attributes.
2. Drag the slider labeled "Shin" to the left or right. Alternatively, type in a value from 0.0 to 1.0 in the numerical entry next to the slider.

Note:

The Specular power changes the size of the highlight. A higher degree will make smaller and sharper highlights, characteristic of shiny objects.

Example:

- Select a simple object and render it in full color with default coefficients, but make sure you select "Phong" from the shaders on the right side of the screen.
- Bring up the Attr menu and drag the "Shin" slider to the right. Render the object again. This time, the object should have smaller and sharper highlights.



Purpose:

To set the specular reflectance of a material.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Type in a value from 0.0 to 1.0 in the "Spec" field.

Note:

The specular coefficient describes how much of the incoming light will be reflected directly from the surface of an object toward the viewpoint. Objects with a high degree of specular reflectance, such as shiny plastic, will exhibit highlights. This highlight takes on the color of the incoming light, rather than the color of the surface.

In order to see the effect of the change in the ambient coefficient, you must render the object.

The value for the specular coefficient is indirectly proportional to the diffuse value, i.e. as the diffuse value increases to 1.0 the specular value decreases to 0.0

Example:

- Select a simple object and render it in full color with default coefficients, but make sure you select "Phong" from the shaders on the right side of the screen.
- Bring up the Attr menu and type 1.0 into the Spec field. Render the object again. This time the object should have more pronounced highlights.

Note: when entering numerical values, you are not restricted to the range of 0.0 to 1.0. Traditionally the sum of the Ambi, Diff and Spec values total 1.0, but we do not impose this restriction. Feel free to experiment with values outside this range, but we do not recommend that you exceed a total of 2.0 because you will begin to intro-

duce artifacts and generally undesirable effects into the image. If the Ambi, Diff, Spec total does exceed 1.0 and you then use a slider to change one of the values, they will ALL be recomputed to bring the total back to 1.0.

3.5.2 Textures

The ability to map an image onto the surface of an object is essential in an effort to render realistic images. Caligari enables the user to map any size Targa .WIN (uncompressed) or Caligari .6rn image onto a variety of surfaces.

Preparation of textures.

The first step in texture mapping is creating a texture. This process starts with painting, digitizing or rendering a 24/32 bit image. Caligari directly accepts two image formats for conversion to a texture: .6rn (Rendition Caligari native format) and .WIN (Truevision Targa/Vista PC format). Some paint programs, such as Macro Paint 24 and TV Paint will generate .6rn format images directly. In addition, there are several third party products such as Raster Link from Active Circuits, inc. and The Art Department from ASDG, inc. that provide a means to convert IFF24 (and other format) pictures into .6rn files.

Once the user has the image in .6rn format, he/she is ready to convert it to a texture. The user enters the attribute menu in the Scene Composition Brender section and from there selects the Texture Controls submenu. A menu with a large rectangle appears. At the right side of the menu is a group of buttons used to select the format of the image to convert to a texture and the name of the texture which has been loaded or is to be created. If the .6rn/.WIN file is on a PC Bridgeboard partition, then the "Bridge" button must be selected. If not, then "Amiga" should be highlighted.

From this menu the user must first select a name for the new texture. Than the user selects the .6rn (or TGA) picture file, followed by "make" to create a texture. The newly created texture will be saved into the "textures" directory and will be available for actual mapping onto the 3D object.

TGA format:

In addition to native Amiga formats an IBM PC compatible format (TGA) can be utilized as a source for texture maps.

Usually TGA pictures are generated on Targa using either an abbreviated or a full version of TIPS (or other paint program). With TIPS, an image can be grabbed from any live video signal in real time or painted using the rich collection of paint tools TIPS offers to the user.

Once an image is created, it can be saved using the "Save Window" option (Do not use Csave!).

Parts of any texture can be made selectively transparent simply by painting with color "0,0,0" (black) in TIPS. Remember also that a full-size Targa screen can take up to 1Mb of RAM. Thus the user should not use too many large size textures.

Placement of Textures.

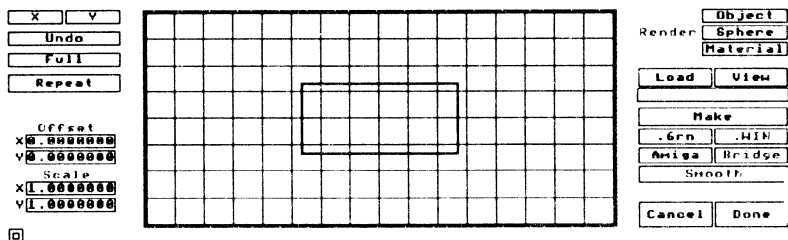
In addition to traditional "mapped" (decals) type of texture placement a new way of "projecting" the texture "through" an object has been implemented. As this is very recent addition to Caligari we refer to the documentation on the program disk itself, which describes the latest changes to the program.

Once there are a few textures in the texture directory, the user can start assigning them to the objects in the scene. After entering the scene module, selecting Brender and the Attrib menu, the user will pick an object to assign the texture to.

NOTE: At present, Caligari can only place a "wrapped" texture on objects which were created with the "Flat" or "Lathe" option of the extruder. Furthermore, Caligari will not texture-map an object created with "Lathe" specifying less than 360 degrees of revolution. Also, if Lathe is used, the object must be rotated around one of its edges and not around an outside axis ("Projected" texture maps have no such limitation).

The Attrib menu will display all possible settings for the current material of the current object. As the user scrolls through all the materials, he can immediately see which parts of the object are colored with that material.

When "Texture" and "Controls" are selected, the user is presented with unwrapped representation of the current object in the form of a grid. The user can load a texture, position the texture interactively by dragging and scaling orange rectangle about the grid.



Even though the interactive texture placement is not done in 3D, the new version is substantially more intuitive than the old numerical entry.

In particular the user can see the polygonal mesh for the object and align the texture rectangle exactly on the edge of a particular polygon. He can also turn on and off X or Y axes and lock the movement of texture rectangle vertically or horizontally.

Numerical values:

The old numerical entry for texture coefficients is preserved. This is the meaning of the variable names:

XOff - Horizontal Offset . . . **YOff** - Vertical Offset

XScale - Horizontal Scaling . **YScale** - Vertical Scaling

Repeat - Repeat of the texture across the object

After setting the values for the placement, the user exits this menu and returns to the Attribute menu. The "Texture" button must be selected. At this point, the user can select "render object" or "render" and see the final result in full color on the frame buffer screen.

Textures can be combined with other rendering attributes, including environment maps. Also a texture can be selectively transparent. For example, a texture with a transparent circle in the center (painted in black with TIPS) could be mapped onto a sphere using the "Repeat" option. If the sphere itself is made transparent, an interesting spherical object with circular windows on it will be rendered.

Also new is the ability to texture map with the Flat shading option (no lights) in addition to the other shaders.

3.5.3 Environment Mapping.

Environment mapping enables the shiny objects in the scene to reflect their environment. A simple form of environment mapping is 1D environment mapping.

1D mapping creates the effect of a horizon reflecting on the object's shiny surface. Caligari has several different 1D maps to choose from.

Cube mapping will actually reflect the existing scene in the selected object. In order to do that, the user must first make a cubic environment and then assign it to the object.



Purpose:

To assign a 1D environment to a selected material attribute.

Procedure:

1. Select an object from the attribute menu. Select one of its material attributes.
2. Select "1D" from the "Environ" group.
3. Select "Load Env" and then select 1D environment from the environment directory. Select "Phong" or "Metal" from the "Shader" group.

Note:

At present, the user can not design his own 1D environments, only select one of the available in the "Load Env" menu. Note also that a 1D (and cubic) environment can be combined with texture mapping.

Purpose:

To assign a cubic environment to a selected material attribute.

Procedure:

1. Select an object from the attribute menu. Select one of its material attributes.
2. Select "Cubic" from the "Environ" group.
3. Select "Phong" or "Metal" from "Shader" group.

Note:

Note also that a cubic (and 1D) environment can be combined with texture mapping.

Purpose:

To create a cubic environment map.

Procedure:

1. Select an object from the attribute menu.
2. Select "Make Cube" from the "Environ" group.
3. Caligari will generate six views of the current scene and save them as a new cubic environment map.

Note:

"Make Cube" will render six views of the current scene, putting the viewpoint into the center of the selected object. Then it will create an environment cube to be used for mapping the environment on the objects selected for cubic mapping.

Since there is only one environment cube, created from the point of view of one particular object, cubic mapping on other objects will produce distorted reflections. Depending on the composition of the scene, this may or may not be acceptable.

Tip:

It is possible to create artificial environments simply by painting (or grabbing) an image and texture mapping it onto a sphere or cylinder (or other object). Make the texture mapped object transparent and insert a small object (cube) inside it. Select the small object and select "Make Cube". This way it is easy to design environment reflections from "cloudy sky" etc.

3.5.4 Surfaces

Caligari objects are constructed from polyhedra, which, in turn, are composed of polygons. Polygons, of course have flat surfaces, creating facet-like appearances on the object. The user can make objects out of very small polygons, thus smoothing the appearance of the object. However, in Caligari Pro there is no way to eliminate facet surfaces completely.

Caligari Broadcast has no such restriction and can create totally smooth surfaces. The program blends the transitions between the polygons, so a dodecahedron will look like a sphere; only the silhouette will give away the true representation of the object.

However, there are times when the user wants the dodecahedron to look like a dodecahedron, or the cube to look like a cube, instead of a poor approximation of a sphere. Caligari allows the user to specify just that.

On the right side of the attribute menu is a rolling button with four positions: **Smooth**, **Complex Fct**, **Simple Fct** and **Auto Fct**. The user can specify any of the four choices for the selected object.

Purpose:

To make a material of the object appear smooth.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. If the "Smooth" option is not already selected, select it now.

Note:

Smooth option will attempt to blend individual polygons into one seemingly continuous surface. It works well if the polygons were designed to approximate the object without sharp edges such as sphere.

The appearance of an object will change, based on the shader selected (see shaders).

Example:

- Design a simple sphere-like object using the "Lathe" option in the extruder. Load it into the scene.
- Select the attribute menu. The default setting for the surface is "Smooth", the default shader is "Gouraud". Leave "Smooth" on and change "Gouraud" to "Phong".
- Select "Render Object" and wait for the rendering to appear on the full color screen. The object will have a smooth appearance.

Purpose:

To make the surface of the object appear faceted.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Select the "Simple Fct" option from the "Surface" group.

Note:

"Simple Fct" option will assign only one color for each polygon, thus creating a faceted appearance of the object. It is useful for a quick preview of the scene, but it does not support a number of advance rendering options such as local highlights, textures, and shadows.

The appearance of an object will change, based on the shader selected (see shaders).

Example:

- Design a simple sphere-like object using the "Lathe" option in the extruder. Load it into the scene.
- Select attribute menu. The default setting for the surface is "Smooth", the default shader is "Gouraud". Leave "Gouraud" on and change "Smooth" to "Simple Fct".
- Select "Render Obj" and wait for the rendering to appear on the Targa screen. The object will have a faceted appearance.

Purpose:

To make the surface of the object appear faceted.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Select "Complex Fct" option from the "Surface" group.

Note:

"Complex Fct" option will preserve the faceted appearance of the object. Unlike the simpler "Simple Fct", it does support all advance rendering options, including local highlights, textures, and shadows

The appearance of an object will change, based on the shader selected (see shaders).

Example:

- Design a simple sphere-like object using the "Lathe" option in the extruder. Load it into the scene. Add a local light source to the scene, positioned next to the new object.
- Select attribute menu. Select "Complex Fct" and change "Gouraud" to "Phong".
- Select "Render Obj" and wait for the rendering to appear on the full color screen. The object will have a faceted appearance and should exhibit local highlights.
- As an alternative, try to put a texture on a complex faceted object.

Purpose:

To make the surface of the object appear faceted on sharp edges and smooth on curved areas.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Select "Auto Fct" option from the "Surface" group.
3. Optionally, change the default angle for transition between facet and smooth areas.

Note:

Auto Fct will smooth the transition between two polygons if the angle at which they connect is less than a given threshold. Default threshold is 20 degrees and it can be changed by the user. Please feel free to experiment to achieve the best results.

The appearance of an object will change, based on the shader selected (see shaders).

Example:

- Design a simple can-like object using the "Lathe" option in the extruder. Load it into the scene. Add a local light source to the scene, positioned next to the new object.
- Select attribute menu. Select "Auto Fct" and change "Gouraud" to "Phong".
- Select "Render Obj" and wait for the rendering to appear on the full color screen. The top and the bottom of the can should be rendered as complex facets while the sides should be smooth.

3.5.5 Shaders

When light energy falls on the surface of an object, it can be absorbed, reflected, or transmitted. There are different methods, or lighting models, which calculate an approximation of these lighting effects. In Caligari they are called Shaders. There are five of them: **Flat**, **Gouraud**, **Phong**, **Metal**, and **Environmental** shader.

Purpose:

To render an object without any light calculation.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Select the "Flat" option from the "Shaders" group.

Note:

The flat shader implements no lighting model at all and merely copies the polygon's color and transparency to the display. It is useful for fast previewing. It is also useful for preview of the texture placement, since it shows the texture exactly as it was painted by the user in 2D program.

Example:

- Design a simple sphere-like object using the "Lathe" option in the extruder. Load it into the scene.
- Select the Attribute menu. Select "Flat" from the Shaders group.
- Select "Render Object" and wait for the rendering to appear on the full color screen. The object will appear as a flat 2D outline.



Purpose:

To render an object using the Gouraud shading model.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Select the "Gouraud" option from the "Shaders" group.

Note:

Gouraud shading provides a fast method of smoothly shading diffuse (non-shiny) surfaces.

The illumination of a Gouraud-shaded surface is determined by ambient reflection and diffuse reflectance. Reasonable values for ambient and diffuse coefficients are 0.375 and 0.625, respectively.

Example:

- Design a simple sphere-like object using extruders "Lathe" option. Load it into the scene.
- Select the Attribute menu. Select "Gouraud" from the Shaders group.
- Select "Render Object" and wait for the rendering to appear on the full color screen. The object will be rendered with a chalk-like character.

Purpose:

To render an object with the Phong shading model.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Select the "Phong" option from the "Shaders" group.

Note:

Unlike the Gouraud shader, the Phong shader computes an illumination intensity independently for each pixel lying within the surface and produces nice highlights.

In addition to ambient and diffuse coefficients, Phong shading requires a coefficient for specular light and a coefficient for surface shininess. Reasonable values are 0.3, 0.4, 0.3, and 32 respectively.

Example:

- Design a simple sphere-like object using the "Lathe" option in the extruder. Load it into the scene.
- Select attribute menu. Select "Phong" from the Shaders group.
- Select "Render Obj" and wait for the rendering to appear on the full color screen. The object will exhibit a clearly defined specular highlight.

Purpose:

To render an object using the metal shading model

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Select the "Metal" option from the "Shaders" group.

Note:

The Metal shader is used to model highly reflective surfaces, such as steel or glass. Because of this, it does not incorporate calculation for diffuse term. Reasonable values for ambient, specular, and shininess coefficients are 0.35, 0.65, and 12, respectively.

Whereas every shader can render transparent surfaces, only the metal shader will cause the transparency of the surface to fall off near the edge of the object.

Example:

- Design a simple sphere-like object using extruders "Lathe" option. Load it into the scene.
- Select attribute menu. Select "Metal" from the Shaders group.
- Select "Render Object" and wait for the rendering to appear on the full color screen.

Purpose:

To render an object using a simple 1D environment.

Procedure:

1. Bring up the Attrib menu, select an object, then select one of its material attributes.
2. Select the "Environ" option from the "Shaders" group.

Note:

The environment shader is a limited 1-dimensional implementation of "environment mapping". It can approximate surfaces such as chrome or glass.

Example:

- Design a simple sphere-like object using the "Lathe" option in the extruder. Load it into the scene.
- Select attribute menu. Select "Environ" from the Shaders group.
- Select "Render Obj" and wait for the rendering to appear on the full color screen. The object will mirror a simple horizon in itself.

